WHAT IS CLAIMED IS:

1. A microprocessor system for executing a program stream, said microprocessor system comprising:

(a) an instruction fetch unit for fetching instructions from an instruction store and for providing a predetermined plurality of said instructions to an instruction buffer;

(b) an execution unit, coupled to said instruction fetch unit, for executing said plurality of said instructions from said instruction buffer in an out-of-order fashion, said execution unit including a load store unit adapted to make load requests to a memory system out-of-order and store requests in-order, said load store unit having,

(i) an address path adapted to manage a plurality of addresses associated with said plurality of said instructions being executed,

(ii) address collision means for detecting and signaling whether address collisions and write pendings exist between each of said plurality of said instructions, wherein said load store unit performs said load requests if no address collisions and no write pendings are detected,

(iii) a data path for transferring load and/or store data to and from said memory system and said execution unit, said data path configured to align data returned from said memory system to thereby permit data falling on a word boundary to be returned from said memory system to said execution unit in correct alignment.

2. The system of claim 1, wherein said address path includes a plurality of address buffers for storing a high order and a low order byte of said load and/or said store request.

3. The system of claim 1, wherein said load store unit further includes means for making multiple memory requests to memory if said data falls on a word

boundary.

4. The system of claim 1, further comprising a data address functional unit, connected to said load store unit, adapted for calculating addresses for said plurality of instructions.

5. The system of claim 4, a virtual memory unit adapted to provide physical address translations, which are generated from a virtual address, to said execution unit and said load store unit.

6. The system of claim 5, wherein said load store unit must have a physical address from said data address function unit and said virtual memory unit before it can make a memory request.

7. The system of claim 1, wherein said instructions are CISC instructions, and wherein said instruction execution unit further comprises decode means for decoding said CISC instructions into RISC instructions.

8. The system of claim 1, wherein said load store unit further comprises means for merging data received from memory with original contents of a destination register.

9. The system of claim 1, further comprising data lines between said execution unit and said load/store data path for directly transferring load and/or execute data to said data path thereby allowing a subsequent store operation to be performed promptly thereafter.

10. The system of claim 1, further comprising an historical pointer for indicating relative age of an instruction in said instruction buffer.

11. The system of claim 1, wherein said collision means indicates a load dependency by determining if there is an address collisions or a pending store address.

12. The system of claim 1, further comprising means for preventing load bypassing of load instructions that will modify the system state incorrectly.

13. The system of claim 1, further comprising means for snooping the results of a function unit in order to provide store data directly to said load/store data path.

14. The system of claim 1, wherein said load store unit has a separate load/store data path for floating point operations.

15. The system of claim 1, wherein said execution unit includes a register file which contains a plurality of real registers and a plurality of temporary registers.

16. In a RISC superscalar microprocessor having an execution unit adapted to execute a stream of instructions and to issue load instructions out-of-order, a method for managing requests for loads and stores to and from a memory device, the method comprising the steps of:

(1) calculating an address for an instruction selected from an instruction window and transferring said address to a load store unit;

(2) determining whether said instruction involves a load operation, a store operation, an execute operation, or a combination of said load, said store, and said execute operations;

(3) checking, if said instruction has a load operation, for an address collision and for any write pendings, and signaling the outcome of said check;

(4) making a request to said memory device based on a priority scheme and the results of said checking step (3);

(5) receiving requested data from said load operation and/or store operation in a data path portion of said load store unit; and

(6) aligning said requested data if said requested data is unaligned.

17. The method of claim 16, wherein step (1) of issuing includes a step of performing a data dependency check on said instruction.

18. The method of claim 16, further comprising the step of writing the results of said instruction into a preassigned location in a temporary buffer.

19. The method of claim 18, further comprising the step of providing data to said load store unit by bypassing said temporary buffer.

20. The method of claim 16, further comprising the step of making all requests for store operations in program order.

21. The method of claim 16, further comprising the step of preventing load bypassing of load operations that will modify the system state incorrectly.

22. The method of claim 16, further comprises the step of merging data received from memory with the original contents of a destination register.